This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (original) An intermediate object linking method of linking a plurality of intermediate objects to form an executable object, comprising:

an intermediate object linking order forming step which decides linking orders of the plurality of intermediate objects;

a linking processing step which executes linking processes of the plurality of intermediate objects based on the linking orders decided by the intermediate object linking order forming step to get an executable object;

a comparing step which compares program size of the executable objects obtained by the linking processing step with the program size of a executable objects stored in a storing section every time when the linking order is changed;

a storing step for storing the program size of the executable objects and the linking order obtained by the linking processing step in the storing section to update when the program size of the executable objects obtained by the linking processing step is smaller than the program size of the executable objects stored in the storing section at the comparing step; and

a repeating step for changing the linking orders by the intermediate object linking order forming step and executing repeatedly the linking processing step, the comparing step, and the storing step.

2. (original) An intermediate object linking method according to claim 1, wherein the intermediate object linking order forming step decides the linking orders by a permutation algorithm.

3. (original) An intermediate object linking method according to claim 1, wherein the intermediate object linking order forming step decides the linking orders by a genetic algorithm.

4. (currently amended) An intermediate object linking unit for linking a plurality of intermediate objects to form an executable object, comprising:

an intermediate object linking order forming section which decides linking orders of the plurality of intermediate objects;

a linker starting section which executes linking processes of the plurality of intermediate objects based on the linking orders decided by the intermediate object linking order forming section to form an executable object;

a comparing section which compares program size of the executable objects every time when the linking order is changed; a storing section which stores the program size of the executable objects and the linking order; and

a repeating section which changes the linking orders by the intermediate object linking order forming section and

operating repeatedly the linker starting section, the comparing section, and the storing section,

wherein the comparing section compares the program size of the executable objects formed by the linker

starting section with the program size of the executable objects stored in the storing section,

wherein the storing section stores the program size of the executable objects and the ~~inker~~ linking order formed by the linker starting section when the program size of the executable objects formed by the linker starting section is smaller than the program size of the executable objects stored in the storing section at the comparison by the comparing section.

5. (original) An intermediate object linking unit according to claim 4, wherein the intermediate object linking order forming section decides the linking orders by a permutation algorithm.

6. (original) An intermediate object linking unit according to claim 4, wherein the intermediate object linking order forming section decides the linking orders by a genetic algorithm.

7. (currently amended) A linker unit for linking a plurality of intermediate objects to form an executable object, comprising:

an intermediate object linking order forming section which decides linking orders of the plurality of intermediate objects;

a linking processing section which executes linking processes of the plurality of intermediate objects based on the linking orders decided by the intermediate object linking order forming section to form an executable object;

a comparing section which compares program size of the executable objects every time when the linking order is changed; a storing section which stores the

program size of the executable objects and the linking order; and

a repeating section which changes the linking orders by the intermediate object linking order forming section and operating repeatedly the linker starting section, the comparing section, and the storing section,

wherein the comparing section compares the program size of the executable objects formed by the linker starting section with the program size of the executable objects stored in the storing section,

wherein the storing section stores the program size of the executable objects and the ~~inking~~ linking order formed by the linker starting section when the program size of the executable objects formed by the linker starting section is smaller than the program size of the executable objects stored in the storing section at the comparison by the comparing section.

8. (original) A linker unit according to claim 7, wherein the intermediate object linking order forming section decides the linking orders by a permutation algorithm.

9. (original) A linker unit according to claim 7, wherein the intermediate object linking order forming section decides the linking orders by a genetic algorithm.

10. (original) A compiler driving unit for translating a source program by starting a compiler, an assembler, a linker, etc. to form an executable object, comprising:

an intermediate object linking order forming section which decides linking orders of the plurality of intermediate objects;

a linker starting section which executes linking processes of the plurality of intermediate objects based on the linking orders decided by the intermediate object linking order forming section to form an executable object;

a comparing section which compares program size of the executable objects every time when the linking order is changed; a storing section which stores the program size of the executable objects and the linking order; and

a repeating section which changes the linking orders by the intermediate object linking order forming section and operating repeatedly the linker starting section, the comparing section, and the storing section,

wherein the comparing section compares the program size of the executable objects formed by the linker starting section with the program size of the executable objects stored in the storing section,

wherein the storing section stores the program size of the executable objects and the ~~inking~~ linking order formed by the linker starting section when the program size of the executable objects formed by the linker starting section is smaller than the program size of the executable objects stored in the storing section at the comparison by the comparing section.

11. (original) A compiler driving unit according to claim 10, wherein the intermediate object linking order forming section decides the linking orders by a permutation algorithm.

12. (original) A compiler driving unit according to claim 10, wherein the intermediate object linking order forming section decides the linking orders by a genetic algorithm.

13. (original) A recording medium for recording a program for linking a plurality of intermediate objects to form an executable object, wherein the program for causing a computer to execute a method comprising:

an intermediate object linking order forming step which decides linking orders of the plurality of intermediate objects;

a linking processing step which executes linking processes of the plurality of intermediate objects based on the linking orders decided by the intermediate object linking order forming step to get an executable object;

a comparing step which compares program size of the executable objects obtained by the linking processing step with the program size of a executable objects stored in a storing section every time when the linking order is changed;

a storing step for storing the program size of the executable objects and the linking order obtained by the linking processing step in the storing section to update when the program size of the executable objects obtained by the linking processing step is smaller, than the program size of the executable objects stored in the storing section at the comparing step; and

a repeating step for changing the linking orders by the intermediate object linking order forming step and executing repeatedly the linking processing step, the comparing step, and the storing step.

14. (original) A recording medium according to claim 13, wherein the intermediate object linking order forming step decides the linking orders by a permutation algorithm.

15. (original) A recording medium according to claim 13, wherein the intermediate object linking order forming step decides the linking orders by a genetic algorithm.